1 Objectives

This lecture covers some of the different places variables can be kept in memory. By the end of lecture, you should be familiar with

- The function call stack
  - static and dynamic links, and how nested scopes are resolved
  - frame pointer, stack pointer
- how local variables are stored
- how object member variables are stored
- the difference between boxed and unboxed data

2 Announcements

Exams will be returned on Wednesday.

3 Examples

3.1 Nested Scoping

```
1 let rec foo a =
2   let t = 10 + a in
3   let bar b =
4     let u = 20 in a + u + b +
5       t + foo 9
6   let baz c =
7     let v = 30 in a + v + c +
8       t + bar 5
9   in baz 4
```

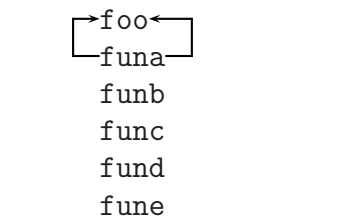
Addr	*Value*
8000	
7996	
7992	
7988	
7984	
7980	
7976	
7972	
7968	
7964	
7960	
7956	
7952	
7948	
7944	
7940	
7936	
...	...

4 Problems

Try the following problems. In a few minutes the instructor will go over the solutions. Feel free to work with the person next to you!

- Consider the code below. The indentation is supposed to show the scope. What does the stack look like if `foo` calls `funa` calls `funb` calls `func` calls `fund` calls `fune`? Just show the static and dynamic links. The links from `funa` to `foo` have been done for you.

```
let foo a = ...
   let funa x = ...
      let funb y = ...
      let func q = ...
   let fund z =
      let fune w = ...
```



- Show the contents of the stack and the heap after the following code is evaluated. Assume that `Leaf` is unboxed.

```
type tree = Branch of int * tree * tree | Leaf
let t = Branch (5, Leaf, Leaf)
let rec u = Branch (7, t, u)
```